EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	3164945	((locat$3 or find$3 or search$3 or quer$3) with ((program$4 near languages) or (program$3 near code)) with type) (database or asset$1 or storage or repositor$3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:18
L2	63	((locat$3 or find$3 or search$3 or quer$3) with ((program$4 near languages) or (program$3 near code)) with type) with (database or asset$1 or storage or repositor$3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:18
L3	0	2 and (language near writ$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:19
L4	16	2 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:19
L5	1	((locat$3 or find$3 or search$3 or quer$3) with ((program$4 near languages near (asset$1 or database)) or (softwar$3 near code near (asset$1 or database))) with type) and database	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:44
S1	2	("6785683").PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:46
S2	1	"6519580".PN.	USPAT; USOCR	OR	OFF	2006/10/12 17:44
S3	1	"6330530".PN.	USPAT; USOCR	OR	OFF	2006/10/12 17:44
S4	435	((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type).ab.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:48

EAST Search History

S5	66	((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type).ti.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:54
S6	13	S5 and S4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:49
S7	7	S6 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:50
S8	21	S4 and (programming with language$1)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:54
S9	6	S8 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:54
S10	790	((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type).clm.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:56
S11	160	S10 and (programming with language$1)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:54
S12	48	S11 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:56

EAST Search History

S13	0	S12 and (analyz$3 with language with type)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:55
S14	0	S4 and (analyz$3 with language with type)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:55
S15	0	S4 and (analyz$3 with language with writ$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:56
S16	6250	((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:02
S17	6	S16 and (analyz$3 with language with writ$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:58
S18	3	S17 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:58
S19	717	S16 and (language with writ$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:08
S20	18707999	19and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 17:58

EAST Search History

S21	232	S19 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:03
S22	2559	((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type) and database	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:26
S23	30	(((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type) and database).ab.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:06
S24	9	S23 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:07
S25	31	(((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type) and (database or asset$1)).ab.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:14
S26	9	S25 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:11
S27	2597	(((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 with languages)) with type) and (database or asset$1))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:07
S28	727	S27 and (code with (asset$1 or storag$3 or database) with type)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:09

EAST Search History

S29	192	S28 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:08
S30	63	S29 and (language with writ$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:09
S31	39	S29 and (language near writ$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:10
S32	0	S27 and (code near (asset$1 or storag$3 or database) near type)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:10
S33	523	S27 and (code near type)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:10
S34	18	S33 and (language near writ$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:19
S35	11	S34 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:15
S36	0	(((locat$3 or find$3 or search$3 or quer$3 or retriev$3) near (code or (program$4 with languages)) near type) and (database or asset$1)).ab.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:14

EAST Search History

S37	20	S33 and (707/3).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:15
S38	12	S33 and (707/10).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:15
S39	8	S38 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:17
S40	7	S37 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:19
S41	0	S29 and (717/8).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:20
S42	0	S29 and (717/168).ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:20
S43	2552	((locat$3 or find$3 or search$3 or quer$3) with (code or (program$4 near languages) or (program$3 near code)) with type) and database	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:26
S44	126	((locat$3 or find$3 or search$3 or quer$3) with ((program$4 near languages) or (program$3 near code)) with type) and database	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/13 10:43

EAST Search History

S45	23	S44 and @ad<"19991229"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/12 18:27

Sign in

Web Images Video[New!] News Maps more »

"code assets" locating "programming language Search Advanced Search Preferences

Web Results **1** - **10** of about **67** for **"code assets" locating "programming languages"**. (**0.24** seconds)

Sponsored Links

Asset investigators
The asset search firm trusted by major law firms around the world.
www.checkmatereports.com

Asset Searches
Bank Accounts, Investments And More For Help Call Now 1-800-250-8885
www.assetsearches.com

Koders - Source Code Search Engine
The Workgroup Edition indexes over 30 **programming languages** and connects to ... and securely index a company's proprietary software source **code assets**. ...
www.koders.com/info.aspx?page=EnterpriseInfo - Similar pages

SERC Technical Reports
The technique should be particularly valuable for **code assets** that will be ... pseudocode that captures key structures of modern **programming languages**. ...
www.serc.net/library/publication/reportByAuthor.asp?AuthorName=Norman%20Wilde - 40k - Cached - Similar pages

SERC Technical Reports
The approach can also help in designing new test cases and in **locating** errors ... pseudocode that captures key structures of modern **programming languages**. ...
www.serc.net/library/publication/reportByCategory.asp?Category=Software%20Maintenance%20 - 61k - Cached - Similar pages

FOCAL project home page
Evolving Legacy Systems by **Locating** System Features using Regression Test ... Legacy systems are now commonly written in modern **programming languages** and ...
www.cs.wpi.edu/~heineman/projects/focal.html - 9k - Cached - Similar pages

DACS SEBD Search Results - Software Reuse
In many concurrent **programming languages** programs are difficult to extend and ... Particularly difficult problems in reusing trusted source **code assets**, ...
www.thedacs.com/topics/cetm/reuse.shtml - 88k - Cached - Similar pages

The Amos Project: An Approach to Reusing Open Source Software
describing capabilities of open source **code assets**, and ... minimizing the number of **programming languages** in the final set of packages; minimizing the ...
es.tldp.org/Presentaciones/200211hispalinux/mcarro/amos-hispalinux-html/ - 45k - Cached - Similar pages

Categorization and presentation tool for code resources - Patent ...
A computer-implemented method for **locating** program **code assets** stored on a storage ... **programming languages**, in a distributed or network environment. ...
www.freepatentsonline.com/6785683.html - 84k - Cached - Similar pages

Put Modern Code Generation to Work
Language: The number of **programming languages** in use is large, but most of our ... the code generator adds value by creating a repository of **code assets**, ...
www.devx.com/enterprise/Article/17550/1954?pf=true - 25k - Cached - Similar pages

[PDF] REUSE OF PERSONAL SOFTWARE ASSETS:
File Format: PDF/Adobe Acrobat - View as HTML
Source **code assets** can thus be located on this site relatively quickly, assuming ... Additionally, many developers use multiple **programming languages** to ...

etd.lib.fsu.edu/theses/available/etd-08282003-183319/unrestricted/01_rjn_thesis.pdf -
Similar pages

[PDF] Design Intent in an Agile Context
File Format: PDF/Adobe Acrobat - View as HTML
Bind **code assets** to tests (reverse traceability) in. order to facilitate propagation of ...
Traditional **programming languages**. Domain specific abstractions ...
www.ece.utexas.edu/~perry/education/382v-s06/L11.pdf - Similar pages

Gooooooogle ▶
Result Page: **1** 2 3 4 5 6 7 **Next**

"code assets" locating "programming Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

IEEE Xplore® RELEASE 2.1

Search Results

BROWSE SEARCH IEEE XPLORE GUIDE

Results for **"('code assets' locating 'programming languages'<in>metadata)"**

e-mail

Your search matched **0** documents.

A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

('code assets' locating 'programming languages'<in>metadata) Search

☐ Check to search only within this results set

Display Format: ◉ Citation ○ Citation & Abstract

» Key

	Indicates full text access
IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help pages if you need assistan search.

Terms used code assets locating programming languages

Found **15,723** of **186,844**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown
Relevance scale

1 Developing software for large-scale reuse (panel)
Ed Seidewitz, Brad Balfour, Sam S. Adams, David M. Wade, Brad Cox
October 1993 **ACM SIGPLAN Notices , Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications OOPSLA '93**, Volume 28 Issue 10
Publisher: ACM Press
Full text available: pdf(735.75 KB) Additional Information: full citation, references, index terms

2 Predicting software quality for reuse certification
William M. Thomas, Deborah A. Cerino
November 1995 **Proceedings of the conference on TRI-Ada '95: Ada's role in global markets: solutions for a changing complex world**
Publisher: ACM Press
Full text available: pdf(1.35 MB) Additional Information: full citation, references

3 Re-engineering legacy Cobol programs
J. K. Joiner, W. T. Tsai
May 1998 **Communications of the ACM**
Publisher: ACM Press
Full text available: pdf(116.24 KB) Additional Information: full citation, references, index terms

4 Ten Mini-Languages: A Study of Topical Issues in Programming Languages
Henry F. Ledgard
September 1971 **ACM Computing Surveys (CSUR)**, Volume 3 Issue 3
Publisher: ACM Press
Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, references, citings, index terms

The proliferation of programming languages has raised many issues of language design, definition, and implementation. This paper presents a series of ten mini-languages, each of which exposes salient features found in existing programming languages. The value of the mini-languages lies in their brevity of description and the isolation of important

linguistic features: in particular, the notions of assignment, transfer of control, functions, parameter passing, type checking, data structures, ...

5 The denotational semantics of programming languages
R. D. Tennent
August 1976 **Communications of the ACM**, Volume 19 Issue 8
Publisher: ACM Press
Full text available: pdf(1.70 MB) Additional Information: full citation, abstract, references, citings

This paper is a tutorial introduction to the theory of programming language semantics developed by D. Scott and C. Strachey. The application of the theory to formal language specification is demonstrated and other applications are surveyed. The first language considered, LOOP, is very elementary and its definition merely introduces the notation and methodology of the approach. Then the semantic concepts of environments, stores, and continuations are introduced to model classes of programmin ...

Keywords: GEDANKEN, LOOP, applicative, continuation, environment, higher-order function, imperative, programming language, recursive definition, semantics, store, theory of computation

6 Technical correspondence: Denotational semantics of programming languages and compiler generation in PowerEpsilon
Ming-Yuan Zhu
September 2001 **ACM SIGPLAN Notices**, Volume 36 Issue 9
Publisher: ACM Press
Full text available: pdf(1.24 MB) Additional Information: full citation, abstract, references, citings, index terms

Programming in constructive type theory corresponds to theorem proving in mathematics: the specification plays the role of the proposition to be proved and the program is obtained from the proof. In this paper, we present an approach of using constructive type theory to derive a compiler of a given programming language from its denotational semantic definition. The development is supported by a proof development system called **PowerEpsilon**.

7 A slicing-based approach for locating type errors
F. Tip, T. B. Dinesh
January 2001 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 10 Issue 1
Publisher: ACM Press
Full text available: pdf(443.36 KB) Additional Information: full citation, abstract, references, citings, index terms, review

The effectiveness of a type-checking tool strongly depends on the accuracy of the positional information that is associated with type errors. We present an approach where the location associated with an error message e is defined as a slice Pe of the program P being type-checked. We show that this approach yields highly accurate positional information: Pe is a progr ...

Keywords: abstract interpretation, program slicing, semantics-based tool generation, static semantics, type-checking

8 A case study in specifying the semantics of a programming language
Ravi Sethi
January 1980 **Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles**

of programming languages
Publisher: ACM Press
Full text available: pdf(904.87 KB) Additional Information: full citation, abstract, references, citings

On and off over the period of about a year I have worked on a semantic specification for the C programming language My objective was to construct a readable and precise specification of C, aimed at compiler writers, maintainers, and language pundits. This paper is a report on the project.

9 Reusable software components
Trudy Levine
July 1998 **ACM SIGAda Ada Letters**, Volume XVIII Issue 4
Publisher: ACM Press
Full text available: pdf(897.86 KB) Additional Information: full citation, index terms

10 Towards bridging the gap between programming languages and partial evaluation
Anne-Françoise Le Meur, Julia L. Lawall, Charles Consel
January 2002 **ACM SIGPLAN Notices , Proceedings of the 2002 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation PEPM '02**, Volume 37 Issue 3
Publisher: ACM Press
Full text available: pdf(209.84 KB) Additional Information: full citation, abstract, references, citings

Partial evaluation is a program-transformation technique that automatically specializes a program with respect to user-supplied invariants. Despite successful applications in areas such as graphics, operating systems, and software engineering, partial evaluators have yet to achieve widespread use. One reason is the difficulty of adequately describing specialization opportunities. Indeed, under-specialization or over-specialization often occurs, without any direct feedback to the user as to the s ...

11 Information structure models: Data structure models for programming languages
Peter Wegner
February 1971 **ACM SIGPLAN Notices**, Volume 6 Issue 2
Publisher: ACM Press
Full text available: pdf(6.62 MB) Additional Information: full citation, abstract, references

This paper introduces a class of models (information structure models) for characterizing computations in terms of the data structures to which they give rise during execution, shows how such models can be used to characterize automata, digital computers and programming languages, considers in some detail the data structures generated during the execution of programs in block structure languages, develops a model for a non-block structure language (SNOBOL 4) and indicates how information structu ...

12 Region-based shape analysis with tracked locations
Brian Hackett, Radu Rugina
January 2005 **ACM SIGPLAN Notices , Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '05**, Volume 40 Issue 1
Publisher: ACM Press
Full text available: pdf(205.67 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper proposes a novel approach to shape analysis: using local reasoning about individual heap locations instead of global reasoning about entire heap abstractions. We

present an inter-procedural shape analysis algorithm for languages with destructive updates. The key feature is a novel memory abstraction that differs from traditional abstractions in two ways. First, we build the shape abstraction and analysis on top of a pointer analysis. Second, we decompose the shape abstraction into a s ...

Keywords: memory leaks, memory management, shape analysis, static error detection

13 Types and persistence in database programming languages
Malcolm P. Atkinson, O. Peter Buneman
June 1987 **ACM Computing Surveys (CSUR)**, Volume 19 Issue 2
Publisher: ACM Press
Full text available: pdf(7.91 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Traditionally, the interface between a programming language and a database has either been through a set of relatively low-level subroutine calls, or it has required some form of embedding of one language in another. Recently, the necessity of integrating database and programming language techniques has received some long-overdue recognition. In response, a number of attempts have been made to construct programming languages with completely integrated database management systems. These lang ...

14 Partial evaluation of high-level imperative programming languages with applications in hard real-time systems
Vivek Nirkhe, William Pugh
February 1992 **Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Publisher: ACM Press
Full text available: pdf(1.17 MB) Additional Information: full citation, references, citings, index terms

15 Models of programming languages: Modelling of storage properties of higher level languages
Kurt Walk
February 1971 **ACM SIGPLAN Notices**, Volume 6 Issue 2
Publisher: ACM Press
Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, references

The role of storage in the characterization of higher level programming languages is discussed. Assignment, in particular, has significantly different meaning in different languages, which can hardly be understood without reference to an underlying model of storage. A general storage model is sketched which can be specialized to a model of ALGOL 68 or of PL/I storage. The same model is used to discuss language features allowing highly flexible data structures.

16 On the Complexity of Flowchart and Loop Program Schemes and Programming Languages
H. B. Hunt
January 1982 **Journal of the ACM (JACM)**, Volume 29 Issue 1
Publisher: ACM Press
Full text available: pdf(934.49 KB) Additional Information: full citation, references, citings, index terms

17 Fable: A programming-language solution to IC process automation problems

Harold L. Ossher, Brian K. Reid
June 1983 **Proceedings of the 1983 ACM SIGPLAN symposium on Programming language issues in software systems**
Publisher: ACM Press

Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

The Stanford University Center for Integrated Systems is embarking on an ambitious project to formally characterize integrated circuit fabrication processes, and to provide a degree of automation of research and prototyping activities in the IC fabrication facility. A crucial component of this project is the ability to represent an IC fabrication "recipe" in a repeatable, transportable, device-independent fashion. We have designed the language Fable for this purpose: it offers s ...

18 Discriminative sum types locate the source of type errors
Matthias Neubauer, Peter Thiemann
August 2003 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN international conference on Functional programming ICFP '03**, Volume 38 Issue 9
Publisher: ACM Press

Full text available: pdf(250.07 KB) Additional Information: full citation, abstract, references, citings, index terms

We propose a type system for locating the source of type errors in an applied lambda calculus with ML-style polymorphism. The system is based on discriminative sum types---known from work on soft typing---with annotation subtyping and recursive types. This way, type clashes can be registered in the type for later reporting. The annotations track the potential producers and consumers for each value so that clashes can be traced to their cause.Every term is typeable in our system and type inferenc ...

Keywords: polymorphism, type errors, type inference

19 Programming languages for distributed computing systems
Henri E. Bal, Jennifer G. Steiner, Andrew S. Tanenbaum
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3
Publisher: ACM Press

Full text available: pdf(6.50 MB) Additional Information: full citation, abstract, references, citings, index terms, review

When distributed systems first appeared, they were programmed in traditional sequential languages, usually with the addition of a few library procedures for sending and receiving messages. As distributed applications became more commonplace and more sophisticated, this ad hoc approach became less satisfactory. Researchers all over the world began designing new programming languages specifically for implementing distributed applications. These languages and their history, their underlying pr ...

20 The Atomos transactional programming language
Brian D. Carlstrom, Austen McDonald, Hassan Chafi, JaeWoong Chung, Chi Cao Minh, Christos Kozyrakis, Kunle Olukotun
June 2006 **ACM SIGPLAN Notices , Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation PLDI '06**, Volume 41 Issue 6
Publisher: ACM Press
Full text available: pdf(244.69 KB) Additional Information: full citation, abstract, references, index terms

Atomos is the first programming language with implicit transactions, strong atomicity, and a scalable multiprocessor implementation. Atomos is derived from Java, but replaces its

synchronization and conditional waiting constructs with simpler transactional alternatives.The Atomos watch statement allows programmers to specify fine-grained watch sets used with the Atomos retry conditional waiting statement for efficient transactional conflict-driven wakeup even in transactional memory systems with ...

Keywords: conditional synchronization, java, multiprocessor architecture, transactional memory